Escuela Politécnica Superior

Sistemas basados en EEG para el control de la síntesis del sonido

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:
Ignacio Blasco Muñoz

Tutor:
José Manuel Iñesta Quereda

Universitat d'Alacant
Universidad de Alicante

Septiembre 2019

Sistemas basados en EEG para el control de la síntesis del sonido

Autor

Ignacio Blasco Muñoz

Tutor

José Manuel Iñesta Quereda
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

Alicante, septiembre de 2019

Abstract

Employing Neural Networks as a computational model for developing artificial intelligence is very extended today. The increase in the performance of parallel processing and the investigation of new models have allowed the error margin committed by these systems to be equal and even less than that of a human being.

In addition, these systems can find patterns in exorbitantly large data sets in which the brightest of people could only be lost without reaching a clear conclusion.

Therefore, the number of research groups that use neural networks as a tool for interpreting data is increasing, developing preventive diagnosis systems of degenerative diseases or forecasting the economic evolution of markets.

This Final Degree Project is within this same line, looking for a system capable of controlling a sound synthesizer from the interpretation of brain waves obtained by real-time electroencephalography.

To this end, a system for collecting samples of volunteers will be implemented, from which we will extract the necessary data to subsequently train different neuronal models and analyse the suitability of these.

In case of obtaining favourable results, the adaptation of the system to a simple functional sound synthesizer will be sought.

Resumen

El uso de redes neuronales como modelo computacional para el desarrollo de inteligencias artificiales está a la orden del día. El aumento en las prestaciones de los equipos de trabajo en paralelo y la investigación de nuevos modelos han permitido que el margen de error cometido por estos sistemas sea igual e incluso inferior al de un ser humano.

Además, estos sistemas tienen la capacidad de encontrar patrones en conjuntos de datos desorbitadamente grandes en los que la más brillante de las personas no podría más que perderse sin llegar a una conclusión clara.

Por ello el número de grupos de investigación que utilizan redes neuronales como herramienta para la interpretación de datos está en aumento, elaborando sistemas de diagnóstico preventivo de enfermedades degenerativas o de previsión de la evolución económica de los mercados.

Este Trabajo de Fin de Grado se encuentra dentro de esta misma línea de investigación, buscando un sistema capaz controlar un sintetizador sonoro a partir de la interpretación de ondas cerebrales obtenidas mediante electroencefalografía en tiempo real.

Para ello se implementará un sistema de recopilación de muestras de voluntarios y voluntarias, del que extraeremos los datos necesarios para posteriormente entrenar distintos modelos neuronales y analizar la conveniencia de estos.

Si se diera el caso de obtener resultados favorables se buscará la adaptación del sistema a un sintetizador de sonido sencillo funcional.

Agradecimientos

Antes de nada, me gustaría agradecer a algunas de las personas que me han acompañado durante mi camino y que, aunque algunas hoy ya no formen parte de él, han influenciado mis pasos pasados, presentes y futuros.

Me gustaría empezar agradeciéndole todo su apoyo a José Manuel, mi tutor durante este trabajo, quien habiendo compartido conmigo nada más que una asignatura optativa durante el grado no vaciló en darme cuanto he pedido. Gracias por darme la oportunidad de aprender tanto con las prácticas de empresa en el DLSI y por aceptar y promocionar la idea que acabaría convirtiéndose en este trabajo.

También quiero agradecer a las compañeras y a los compañeros que han compartido su grado conmigo, culpables de que estos cuatro años hayan pasado en un suspiro, que lo que Fourier ha unido, ningún trabajo lo separe.

Así pues, y aunque su trabajo para conmigo acabase en el momento en el que puse un pie dentro de la universidad, no puedo dejar pasar la ocasión de darle las gracias a Lluís, mi profesor de matemáticas en bachiller. Gracias por contagiarme el amor por el desarrollo matemático, sin duda este ha conseguido que disfrute del grado de una forma que sin él jamás lo habría hecho.

Además, quiero agradecerle, otra vez, a Rebeca todo lo que hemos vivido, todo lo que hemos reído y todo lo que hemos disfrutado. Gracias por comprender y compartir conmigo casi todas mis rarezas. Te quiero.

Y, por último, y por ello más importante, agradecerle a mi familia, a Jalo, Kiquelo, papá y mamá por darme siempre todo lo que han tenido, por haberme hecho sentir el más grande, a pesar de ser el más pequeño. Por haberme visto y hecho crecer, desde la cuna hasta hoy. Por admirarme y apoyarme en mi dedicación a lo que anhelo, la ingeniería.

A todas y a todos, gracias.

A ella, motor de mi vida, fuente de mis ideas. Mi inspiración.

El pensamiento no es más que un relámpago en medio de una noche larga.
Pero ese relámpago lo es ***todo***.

Jules Henri Poincaré.

Índice general

1. Introducción

A lo largo este trabajo fin de grado se abordarán distintos aspectos relacionados con su desarrollo. Para empezar, se especificará qué se espera de él y el estado actual de la investigación en el campo, posteriormente se explicarán algunos de los conceptos teóricos más importantes, se detallará cómo ha transcurrido el proceso experimental, si los resultados son positivos se describirá el sistema final al que se ha llegado y finalmente se hará reflexión sobre el trabajo y las posibles vías de expansión de este.

1.1. Objetivos

El objetivo fundamental del trabajo es el estudio de la viabilidad de un sistema capaz de controlar la síntesis del sonido a través del pensamiento, gracias a la lectura en tiempo real del electroencefalograma del usuario.

Al tratarse de un objetivo tan complejo, es necesario dividirlo en una serie de tareas más concretas, que permitan el pivotaje entre ellas por si fuera necesario modificar alguna de las estrategias escogidas:

1. **Estudio previo:** Se estudiarán las especificaciones técnicas que nos ofrece el equipo de electroencefalografía escogido para el proyecto, además de la forma en la que pueden obtenerse las lecturas en crudo del dispositivo.

2. **Elaboración de interfaz cerebro-ordenador:** Se implementará el subsistema que permita la comunicación entre el dispositivo EEG y las aplicaciones finales.

3. **Toma de datos:** Se diseñará el entorno de toma de datos para el posterior entrenamiento de la red neuronal, y se llamará a las personas voluntarias para la obtención y almacenamiento de las muestras.

4. **Estudio de los modelos neuronales:** Se estudiarán distintos modelos de entrenamiento, variando además la forma en la que se tratan los datos que alimentan a la red.

5. **Desarrollo del sintetizador sonoro (si es viable):** Se desarrollará un sistema sintetizador que sea capaz de ejecutar un conjunto reducido de instrucciones que se llamen desde la red neuronal.

1.2. Estado del arte

Desde hace ya unos años, con el considerable aumento de la capacidad de procesamiento de los equipos informáticos, destacando las tarjetas gráficas cuyos procesadores se especializan en la computación en paralelo, contando algunos de ellos con hasta 4700 núcleos independientes, se ha conseguido que el desarrollo de las redes neuronales, que llevaba décadas estancado, coja más fuerza que nunca, consiguiendo encontrar soluciones que hasta la fecha eran prácticamente imposibles de alcanzar con la algoritmia tradicional.

Entre los grandes logros de esta revolución en el desarrollo de inteligencias artificiales se encuentra AlphaZero, una red que con tan solo un día de entrenamiento fue capaz de derrotar a los campeones mundiales de ajedrez, shogi y go, sin haberle facilitado ningún registro de jugadas o estrategias, simplemente conociendo las reglas del juego.

Por otra parte, gran parte de las tecnologías disponibles de reconocimiento de voz e imagen, como Google Lens o Amazon Alexa son posibles gracias a el entrenamiento de redes neuronales y la inconmensurable cantidad de datos que estas empresas poseen para ello.

Además, esta tecnología no se limita al análisis o clasificación de patrones, el paradigma del aprendizaje profundo se está llevando a lo creativo, con aplicaciones capaces de generar caras humanas que no existen, guiones de cine o componer piezas musicales.

Y de forma más relacionada con la línea de este trabajo, recientemente en la Universidad técnica de Dinamarca han conseguido controlar el vuelo de un dron mediante la lectura del electroencefalograma de diez sujetos de prueba distintos.

Estos y otros avances llevados a cabo en el campo de la inteligencia artificial dejan entrever que el futuro de la tecnología está estrechamente ligado a este, ya que, aunque se trate de una tecnología que se encuentra todavía en fase madurativa, la cantidad de avances logrados gracias al uso de redes neuronales artificiales ha provocado que se estén retomando problemas pendientes de resolución que se consideraban imposibles hasta la fecha.

2. Marco teórico

Los aspectos en los que se basa este trabajo abarcan disciplinas de diversas ramas del conocimiento, como neurofisiología, matemáticas, algoritmia, teoría de la señal y síntesis del sonido.

Sin ánimo de profundizar demasiado en ninguna de estas, se explicarán algunos de los aspectos básicos que se consideran fundamentales para la comprensión del documento.

2.1. Electroencefalografía

Nuestro cerebro, el órgano encargado de controlar el sistema nervioso de nuestro cuerpo, está formado por alrededor de cien millones de neuronas, que se comunican entre sí mediante un proceso que se conoce como sinapsis. Esta sinapsis es un proceso químico que origina una corriente eléctrica de muy poca intensidad.

En 1924 el psiquiatra Hans Berger empezó a registrar esta actividad cerebral en humanos, mediante la técnica que se conoce como electroencefalografía, que consiste en la colocación de unos electrodos sobre la superficie de la cabeza con la sensibilidad suficiente como para medir voltajes de microvoltios.

La colocación de estos electrodos se conoce como Sistema 10-20, ya que los electrodos se sitúan con una separación entre sí de entre el 10 y el 20% de la distancia total entre los puntos más lejanos de la superficie del cráneo.

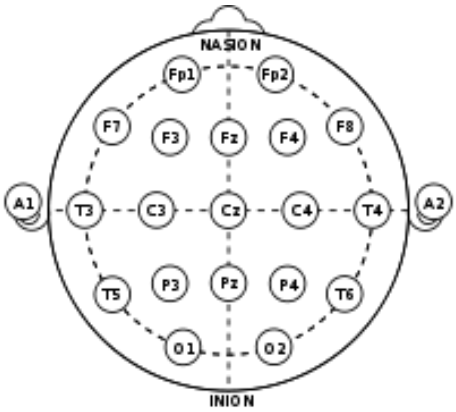


Figura 1. Disposición de los electrodos en el Sistema 10-20.

2.1.1. Tipos de ondas cerebrales

Dentro de las ondas cerebrales que pueden observarse mediante la electroencefalografía pueden distinguirse cinco tipos, en función del rango de frecuencias en que abarcan, que suelen relacionarse con ciertos procesos cerebrales:

Ondas Delta

Estas ondas oscilan con frecuencias menores a los 4 Hz, su amplitud es mayor a la del resto de ondas y predominan sobre las demás cuando estamos durmiendo profundamente.

Ondas Theta

Estas ondas oscilan con frecuencias entre los 4 y los 8 Hz, suelen aparecer en las primeras etapas del sueño, estudios recientes aseguran que están relacionadas con la memoria a corto plazo.

Ondas Alpha

La frecuencia de oscilación de estas ondas está comprendida entre los 8 y 14 Hz, su actividad es mayor cuando se tienen los ojos cerrados en un estado de relajación, por lo que se cree que son producidas por la corteza visual en estado de reposo.

Ondas Beta

Estas ondas tienen una frecuencia comprendida entre los 14 y los 31 Hz, y se asocian al estado de consciencia, pensamiento y concentración activos.

Ondas Gamma

Estas ondas suelen oscilar con frecuencias mayores a los 32 Hz, se relacionan con un alto nivel de actividad cerebral y con la consolidación de información nueva.

2.2. Redes neuronales artificiales

Las redes neuronales artificiales constituyen un modelo matemático que tiene como objetivo el desarrollo de inteligencias artificiales basándose, desde un punto de vista conceptual, en el funcionamiento del cerebro humano.

Este modelo postula que la información que nuestro cerebro almacena en forma de conocimientos, habilidades o hábitos, se manifiesta físicamente en la robustez de las conexiones que se generan entre unas neuronas y otras.

De hecho, la forma en la que se representan las redes neuronales reales y las artificiales son bastante similares:

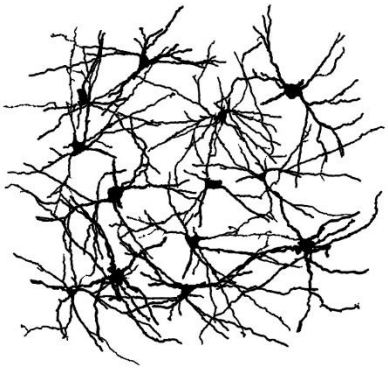


Figura 2. Red neuronal real

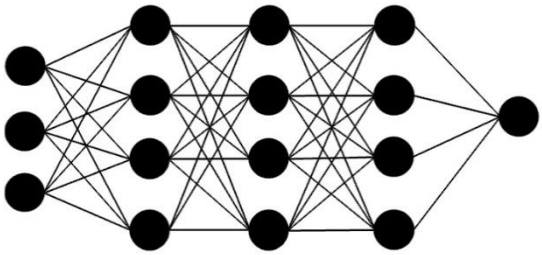


Figura 3. Red neuronal artificial

De esta forma, a modo anticipativo, el paradigma de las redes neuronales consiste en una serie de objetos a los que llamamos neuronas conectados entre sí de forma que, variando la fuerza de cada conexión entre ellas, consigamos que nuestra red presente el comportamiento deseado.

2.2.1. Funcionamiento de una red neuronal

Como puede observarse en la figura 3, las redes neuronales están formadas por capas de neuronas. En este caso se representan de izquierda hacia derecha, siendo la primera capa la capa de entrada, por donde, como su nombre indica, se introducen los datos a tratar por la red. Entre medias se sitúan las capas ocultas y al final la capa de salida, que es la que muestra el resultado de todo el proceso llevado a cabo.

Las neuronas se conectan mediante una unión sináptica artificial a todas las neuronas de las capas adyacentes, son estas uniones en las que, como se especula que ocurre con el

cerebro humano, se almacenará la información necesaria para desarrollar una habilidad en concreto, a partir del entrenamiento.

El papel que juega cada neurona dentro de la red es bastante sencillo, empezaremos a describirlo desde el principio de la red (parte izquierda).

Cada neurona guarda un número, y en el caso de las neuronas de entrada estos números será proporcionados por el usuario, como pueden ser los píxeles de una foto o las muestras de una grabación de voz. En el caso de las neuronas que hay en las demás capas, tanto intermedias como de salida, este número se calcula a base de multiplicaciones y sumas.

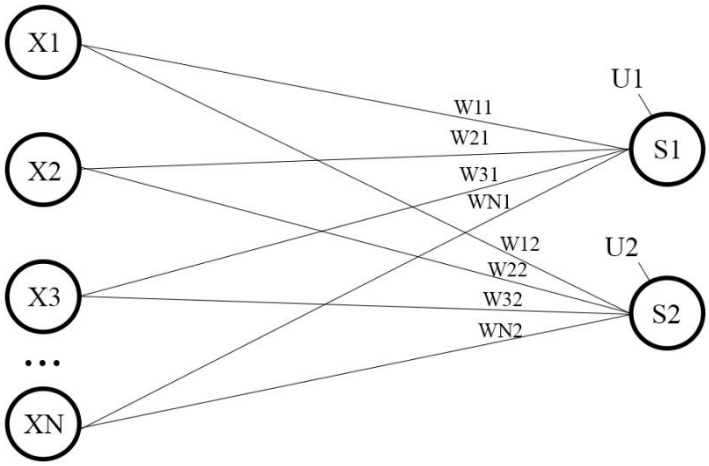


Figura 4. Esquema de unión entre dos capas

Como sabemos, cada neurona está conectada a las neuronas de la capa anterior mediante los enlaces sinápticos, enlaces que poseen de forma individual un número al que llamaremos peso. De esta forma, lo primero que hace una neurona es multiplicar el número de cada neurona de la capa anterior por el peso de la unión que las une, y sumar todos estos productos, además de sumarle a este resultado un parámetro de ajuste U, propio de la neurona. La suma que hace cada neurona de una capa puede expresarse de de la siguiente forma:

$$Sj = Uj + \sum_{i=1}^{N} wij * Xi$$

Fórmula 1. Sumatorio de las neuronas de una misma capa.

Para terminar, una función de activación se encarga de transformar el valor obtenido de la suma y transformarlo en el valor final de la neurona, cuanto mayor sea ese valor, más activada estará la neurona.

Existen muchas funciones de activación distintas, y cada una suele utilizarse para tareas concretas en función de su comportamiento:

Sigmoide

$$f(x) = \frac{1}{1 + e^{-x}}$$

Fórmula 2. Función sigmoide.

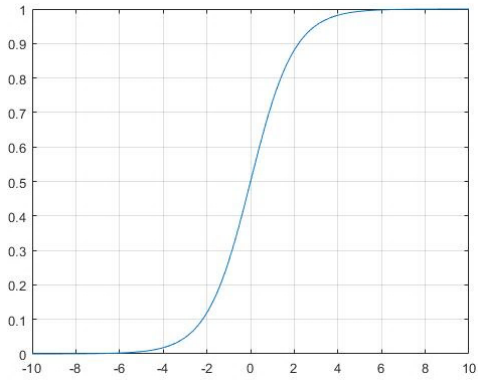


Figura 5. Función sigmoide.

La sigmoide es una función que transforma los valores de una neurona y los acota entre 0 y 1, tendiendo a estos límites de manera asintótica. Se suele utilizar como activación en la última capa de la red.

ReLU – Rectified Lineal Unit

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Fórmula 3. Función ReLU.

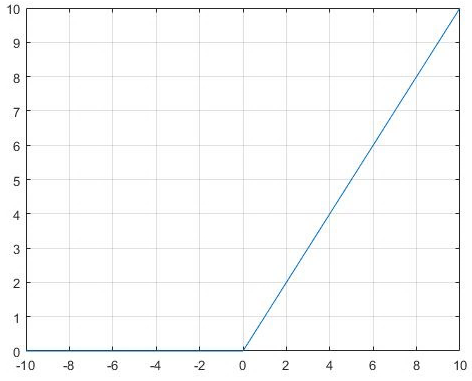


Figura 6. Función ReLU.

La función ReLU es una función que deja el valor que encuentra tal y como está, a no ser que este sea negativo y lo cambie por cero. A diferencia de la función sigmoide, no está acotada superiormente. Se utiliza en redes que trabajan con imágenes.

Softmax

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Fórmula 4. Función softmax.

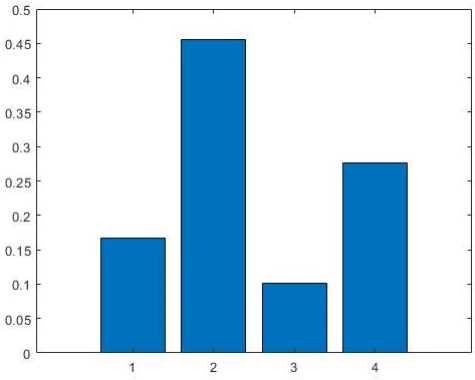


Figura 7. Función softmax.

Esta función consigue que la salida de la red se exprese en términos de distribución de probabilidad, por lo que la suma de todas las salidas debe ser 1. Suele utilizarse en redes cuyo fin es la clasificación.

2.2.2. Entrenamiento de una red neuronal

Una vez explicado el funcionamiento general de una red neuronal pasaremos a explicar cómo es que este artilugio matemático tiene, o simula, la capacidad de aprender de un ser humano.

Previamente hablábamos de que las capas de neuronas estaban unidas entre sí con unas conexiones a las que se le asignaba un peso y que era ese conjunto de valores que asignábamos a los pesos lo que definía el comportamiento de nuestra red.

De esta manera, la forma que tiene una red de aprender es modificar esos pesos, que como al principio no sabemos qué valores han de tener, toman un valor aleatorio.

La forma en la que red ajusta estos pesos es mediante lo que se conoce como entrenamiento, que compara la salida producida por la red con la que se considera correcta, por ejemplo, mediante cálculo del error cuadrático medio. Después, aplica el algoritmo de retropropagación, que consiste en calcular las derivadas parciales del error en función de los distintos pesos de la red y los va ajustando con el objetivo de reducir este error.

Para que la red consiga aprender, se repite este proceso tantas veces como haga falta hasta alcanzar un margen de error aceptable. A cada una de estas repeticiones se le conoce como época.

2.3. Síntesis del sonido

El sonido, tal y como lo conocemos desde el punto de vista de la teoría de la señal, no es más que la suma de un conjunto de sinusoides que oscilan a frecuencias y amplitudes diferentes.

Por norma general, cuando visualizamos la forma de onda de una grabación de sonido, lo que observamos es el desplazamiento de la parte móvil de un micrófono respecto al tiempo, producido por el sonido incidente sobre el mismo.

Aunque existe otra forma de representar el sonido y va en coalición a lo que se ha explicado al principio de este apartado, la representación en frecuencia. La forma de pasar del dominio temporal al dominio frecuencial se conoce como Transformada de Fourier. Esta transformada consigue que la información que tenemos en función del nivel de presión a través del tiempo se convierta en una descomposición de la contribución a una señal de todas las sinusoides de distintas frecuencias.

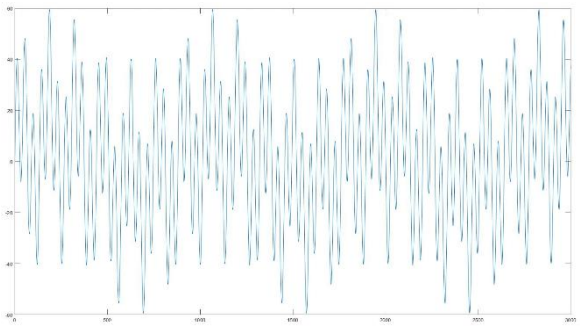


Figura 8. Señal en el dominio del tiempo.

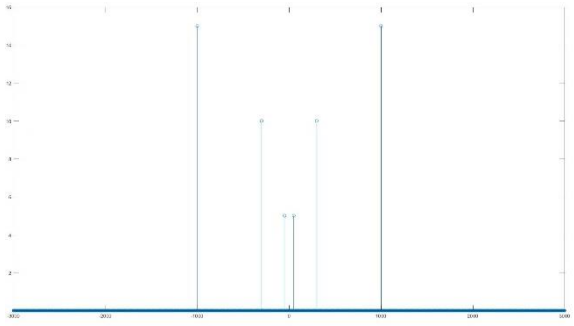


Figura 9. Señal en el dominio de la frecuencia.

Con la representación en frecuencia es mucho más sencillo estudiar cuántas frecuencias componen una señal y en qué medida. En este caso se puede ver que la señal está compuesta por tres sinusoides de distinta frecuencia, la mitad de las que se perciben en la gráfica, ya que la Transformada de Fourier presenta simetría en el origen, mostrando las mismas amplitudes en las frecuencias negativas que en las positivas, que a efectos prácticos son la misma frecuencia. Además, se puede apreciar que la baja frecuencia presenta menor contribución que la media-alta.

La síntesis que producirá nuestro sistema final se limitará a la adición de sinusoides de distintas frecuencias, teniendo en cuenta los aspectos que trataremos a continuación.

2.3.1. Otros aspectos del sonido

Existen multitud de sonidos que se pueden sintetizar, algunos de los cuales pueden encontrarse en la naturaleza y otros que son pura invención humana.

La síntesis creativa presenta un grado de libertad mayor, ya que cuando generamos estos sonidos no tenemos ninguna referencia con los que compararlos y el análisis de la calidad de estos se centra en aspectos más subjetivos.

Por el contrario, cuando intentamos imitar un sonido que existe en la naturaleza, como las notas producidas por un clarinete, nuestro sonido sintético deberá compararse con el sonido real para saber si nuestra síntesis es correcta o no.

En principio las notas musicales son tonos puros, lo que quiere decir que están compuestos por una única sinusoide que vibra a la frecuencia de la nota en cuestión. Pero si eso fuera así no seríamos capaces de distinguir un Do de una flauta travesera al Do de un piano de cola.

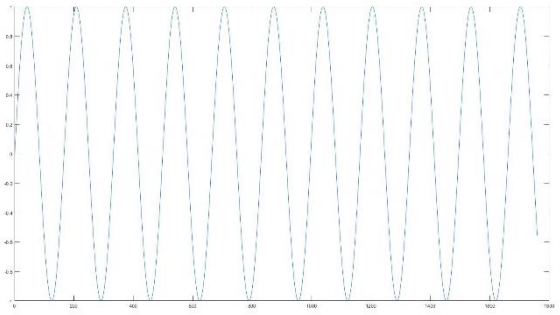


Figura 10. Tono puro Do.

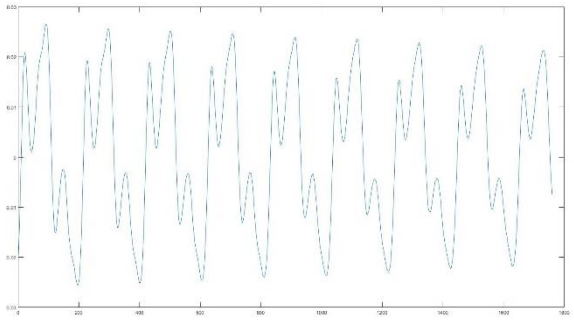


Figura 11. Do de un clarinete.

Lo que distingue a una nota tocada en un instrumento u otro es su composición espectral, y, si bien cuando un instrumento hace sonar una nota, la frecuencia que más predomina en su descomposición espectral, es la de la nota, se pueden apreciar otras frecuencias múltiplos de esta, que se conocen como armónicos, Y al conjunto de los armónicos característicos de cada instrumento se les conoce como timbre.

3. Experimentación

Para empezar con el desarrollo del sistema propuesto tendremos que implementar un sistema que nos permita la recopilación de muestras para el entrenamiento de la red neuronal, y que esta adquiera la capacidad de clasificar nuestros pensamientos en los distintos comandos de nuestro sintetizador.

Ya que no tenemos la certeza de que el sistema que se plantea sea viable con los medios de los que se dispone, haremos que el conjunto de instrucciones sea muy limitado. Solo constará de cuatro posibles comandos: subir, bajar, tono y volumen.

Nuestro sistema tratará de sintetizar notas musicales, las instrucciones tono y volumen seleccionarán los modificadores de frecuencia y amplitud de la onda, respectivamente, y las instrucciones de subir y bajar harán que el modificador seleccionado aumente o disminuya su valor.

Para ello tendremos que hacer que nuestra red sea capaz de distinguir cinco posibles clasificaciones en su lectura del EEG, las cuatro instrucciones y una quinta a la que llamaremos “ruido”.

Por lo tanto, necesitaremos un conjunto de datos formado por muestras etiquetadas como las cinco categorías posibles, que recopilaremos durante diversas sesiones con personas voluntarias. Con el fin de aumentar las probabilidades de éxito, se buscarán personas relacionadas con el Grado en Ingeniería en Sonido e Imagen en Telecomunicación para la toma de muestras, ya que se presupone que están familiarizadas con los conceptos de la teoría de la señal relacionados con la síntesis.

Una vez tengamos las muestras recogidas generaremos otras nuevas a partir de estas añadiendo distintas clases de ruido y degradaciones de forma artificial de manera que la red sea capaz de detectar los patrones en cualquier contexto, independientemente del usuario y de la relación señal a ruido de la transmisión, siempre y cuando esta no sea excesivamente baja.

3.1. Sistema y entorno de pruebas

Para que todas las pruebas se realizaran siempre en el mismo entorno, y que estas estuvieran libres de interferencias por parte de personas ajenas al experimento, decidimos llevarlas a cabo en el laboratorio de sonido del Departamento de Lenguajes y Sistemas Informáticos, con sus equipos de trabajo y una pequeña sala acondicionada como estudio de grabación, en la que se colocará el equipo de toma de datos.

Para llevar a cabo la recogida de datos le mostraremos al sujeto de pruebas una serie de términos en pantalla, entre los cuales se encontrarán las instrucciones de nuestro sintetizador. Mientras tanto, el sensor de EEG irá captando las ondas que se producen en su cerebro y se irán almacenando en una carpeta ya etiquetadas y clasificadas según el nombre de sujeto y fecha.

Además del ruido añadido posteriormente a las muestras para generar otras nuevas, produciremos algunos estímulos visuales y auditivos durante el transcurso de las pruebas, como música y luces de colores, con el fin de generar ruido natural en las ondas cerebrales captadas por el sistema.

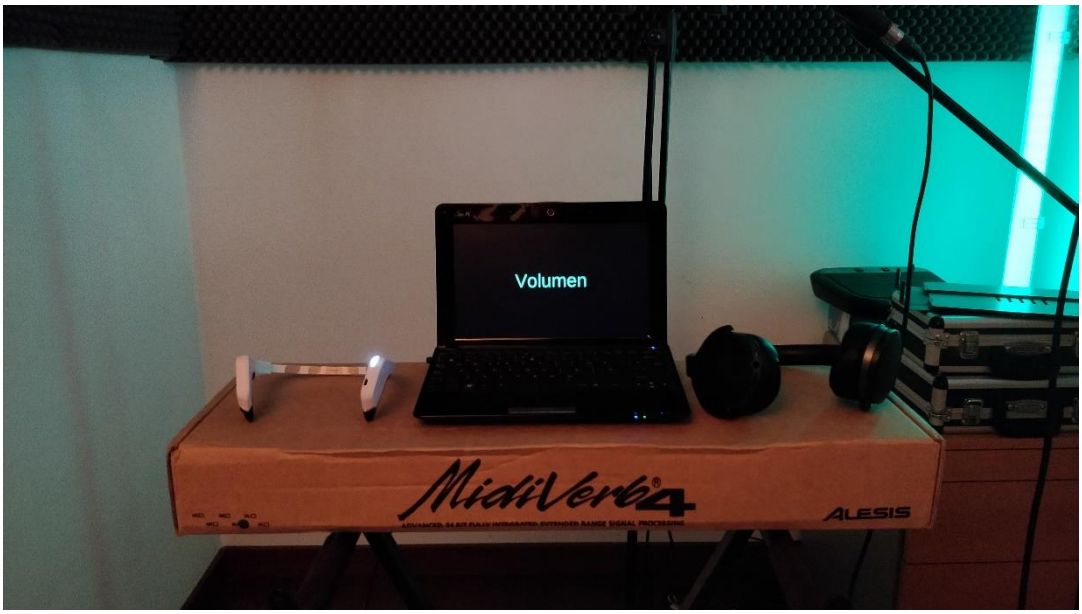


Figura 12. Montaje experimental en el laboratorio.

3.1.1. Hardware escogido

El equipo que utilizaremos para el montaje experimental consta principalmente de dos ordenadores portátiles, uno que se colocará frente al sujeto y otro desde el que se controlará todo el sistema, además de la diadema-sensor de EEG, Muse.

Figura 13. Diadema-sensor Muse.

La diadema Muse dispone de cinco sensores, situados en las posiciones Fpz, AF7, AF8, TP9 y TP10 del Sistema 10-20, de los cuales los tres frontales están fabricados en plata y los dos temporales en una goma de silicona conductora.

Es ajustable desde los 52 centímetros de circunferencia hasta los 60 y su peso es de 61 gramos.

Las frecuencias de muestreo que ofrece son 220 y 500 Hz, y transmite las muestras obtenidas de los cuatro canales del dispositivo mediante conectividad Bluetooth 2.1.

Además, dispone de un acelerómetro de tres ejes con una frecuencia de muestro de 50 Hz, y una resolución de 10 bits, aunque para este estudio no lo tendremos en cuenta.

La gestión de los datos transmitidos se realiza mediante la aplicación MuseDirect, que a su vez es capaz de enviar la información de los sensores de EEG, batería y acelerómetro a otras aplicaciones a través del protocolo OSC, un protocolo de transmisión de datos orientado a transmisión de sonido en tiempo real.

3.1.2. Aplicación de pruebas

Para facilitar el desarrollo tanto de la aplicación de pruebas como la aplicación final y permitir el futuro desarrollo y ampliación de nuevas funciones para el sistema, lo primero que se desarrollará será una librería de Python donde se defina el funcionamiento de algunas funciones que gestionen la conectividad entre el dispositivo EEG y las aplicaciones a implementar.

Esta librería, que de ahora en adelante llamaremos MuseUtils contiene una clase llamada MuseReceiver, cuyos objetos recibirán en el momento de su creación la IP y el puerto del ordenador del que escucharán la transmisión OSC de MuseDirect.

Además de los métodos para conectar y desconectar la escucha OSC, la clase tiene otros métodos destinados a la obtención de la batería actual del dispositivo y a la toma de muestras de EEG.

Los métodos EEGsnap y updateEEG se encargan de crear los ficheros CSV que contengan todas las muestras de los cuatro canales por separado.

Con la librería ya confeccionada, podemos dar paso al desarrollo de la aplicación de toma de muestras, que llamaremos DatasetMaker. Esta aplicación contará con una interfaz gráfica que permita al usuario conectarse al dispositivo Muse y tomar muestras del EEG de la persona que tenga colocada la diadema, para ello tendrá que especificar el nombre del sujeto, el nombre de la muestra y la duración de esta.

Para que la persona a la que se le están realizando las medidas sepa qué tiene que pensar en cada momento se desarrollará una aplicación auxiliar con la ayuda del servicio de mensajería instantánea Telegram, de forma que el ordenador portátil frente al sujeto muestre lo que se introduzca en un chat de Telegram, a no ser que se introduzca el comando /clear, que provocará que en la pantalla del sujeto aparezca una palabra aleatoria para que deje de pensar en la instrucción del sintetizador que se estuviera mostrando.

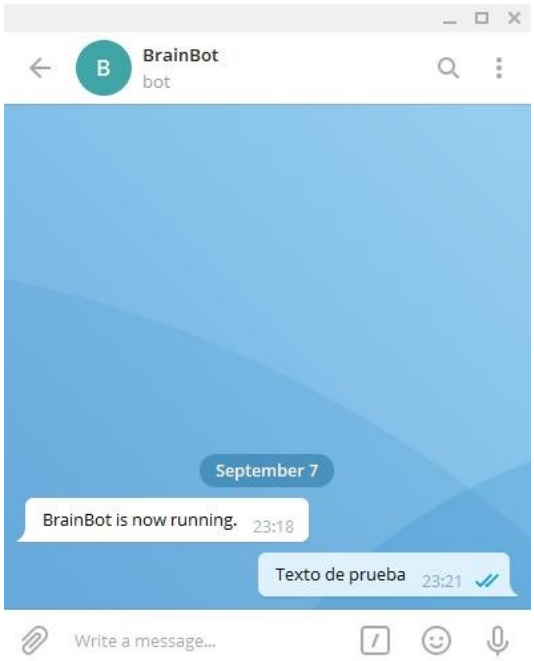


Figura 14. Chat de Telegram.

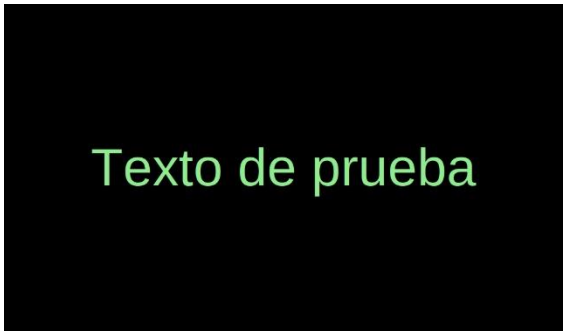


Figura 15. Pantalla que ve el sujeto de pruebas.

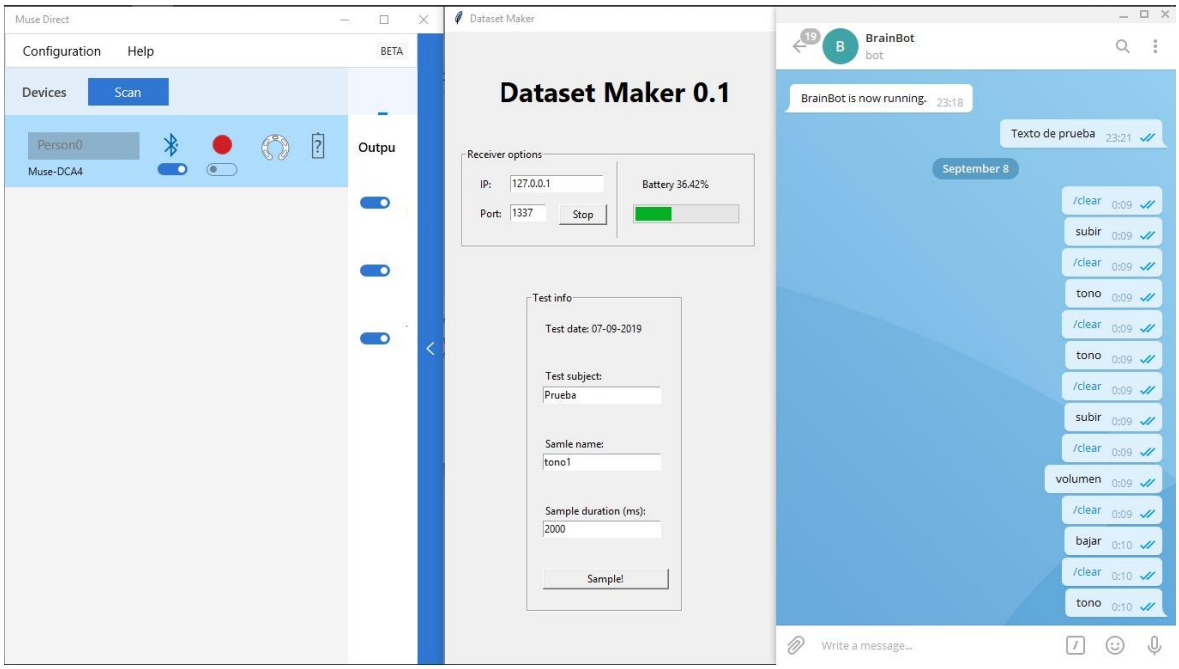


Figura 16. Captura del ordenador que controla el sistema de toma de datos.

3.2. Análisis de los datos recopilados

Los datos tomados durante las sesiones de pruebas fueron recopilados en mediciones de 2000 milisegundos, y como el sistema poseía una frecuencia real de 250 valores por canal y por segundo, el tamaño de las muestras es de 1000 valores en crudo.

Para evitar que el desbalance de las muestras según categoría provocase algún tipo de comportamiento no deseado en nuestra red se realizaron el mismo número de mediciones para cada una de las instrucciones de nuestro sintetizador.

Una vez recogidos los datos de las sesiones de pruebas se procedió al tratamiento de estos a través de MATLAB con el fin de corregir algunos fallos, como valores nulos, mediante la interpolación.

Además, generamos nuevas muestras a partir de las que ya teníamos mediante la adición de ruido blanco gaussiano a estas, de forma que se aumente el tamaño del conjunto de datos con muestras diferentes a las que ya tenemos, pero que pudieran darse en la realidad.

Una vez se han corregido los datos y se han generado las muestras nuevas, procedemos a generar imágenes con los espectrogramas de las muestras y las guardamos para el entrenamiento de las redes. Estos espectrogramas nos dejan ver cómo evoluciona la composición espectral de las muestras en función del tiempo.

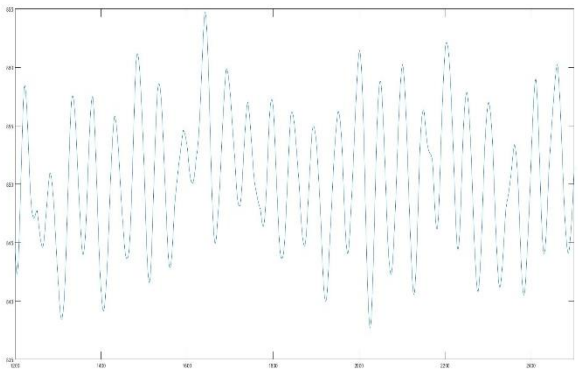


Figura 17. Muestra de EEG en el tiempo.

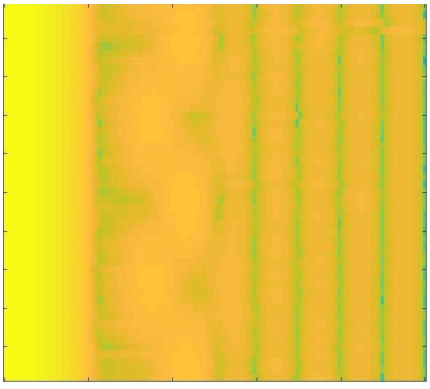


Figura 18. Espectrograma de la muestra.

3.2.1. Valoración de los resultados

Tras entrenar distintos modelos de redes neuronales, tanto de artículos destinados al análisis de EEG, imagen y sonido, no se ha encontrado ninguna alternativa que haga que la propuesta del trabajo sea posible, en la mayoría de los modelos el porcentaje de acierto se quedaba estancado en un 20%, lo que quiere decir que nuestra red tenía el mismo efecto que dar por sentado que el usuario siempre estuviera pensando, por ejemplo, la instrucción “tono”.

Otra casuística que se ha dado es la de una red que memorizaba los espectrogramas de las muestras, lo que se conoce como sobreajuste, alcanzando un acierto del 100% según el entrenamiento, pero que a la hora de la validación no sobrepasaba el 30%, como podemos ver en la siguiente figura:

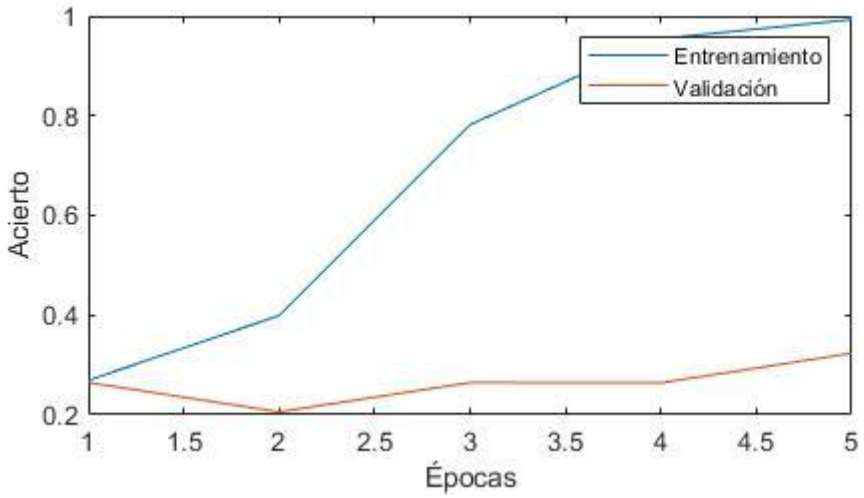


Figura 19. Acierto del entrenamiento frente al de la validación.

Normalmente el fenómeno del sobreajuste se debe a que el conjunto de datos para el entrenamiento no es lo suficientemente grande, por lo que podría ser que de haber realizado más pruebas se hubiera llegado a buen puerto, pero tampoco habría sido garantía de ello.

De hecho, el modelo empelado utilizaba algunas capas de Dropout, cuyo fin es desactivar algunas de las neuronas de forma aleatoria con el fin de evitar este sobreajuste, pero no ha surgido efecto.

4. Conclusiones

Tras analizar los resultados obtenidos del estudio de las muestras, podemos concluir que no es viable implementar un sistema que controle la síntesis del sonido a través de la transcripción pensamiento-instrucción mediante la lectura de un electroencefalograma, aunque no sea en tiempo real. Al menos no bajo estas mismas condiciones, ya que son muchas las razones que podrían provocar que este experimento no presentara los resultados deseados.

Por un lado, podría deberse a que el equipo de medición no tuviera la resolución necesaria, los dispositivos de medición de EEG utilizados en la mayoría de los estudios cuentan con un número de canales muy superior a las de la banda Muse. Por ejemplo, los cascos utilizados en el control del dron mediante la lectura de EEG que se menciona en el apartado del estado del arte cuentan con 64 electrodos, es decir, el volumen de información recopilado en una sola medición es 16 veces mayor. Pero su coste también es mayor, pasaríamos de los 200€ invertidos en este proyecto a 4500€, una inversión más que excesiva para un proyecto de esta envergadura.

Por otra parte, podría deberse a que la cantidad de muestras recopiladas no fuera suficiente y la red pasara de no obtener ningún patrón a memorizar el conjunto entero de datos.

Y, por último, podría ser que las ondas producidas por nuestro cerebro no puedan clasificarse mediante el uso de redes neuronales artificiales, al menos mediante el uso de los modelos empleados.

Sin embargo, antes de considerar el proyecto como un fracaso total, pueden encontrarse números aspectos positivos tras su finalización. Todo el trabajo se ha llevado a cabo teniendo en cuenta la escalabilidad y mejora del sistema, por lo que replantear los aspectos que podrían haber causado que el experimento no tuviera éxito no requiere que se repita todo el esfuerzo, tanto los datos recopilados en las pruebas como el código desarrollado para la obtención de datos es completamente válido tanto para un replanteamiento de este sistema como para cualquier otro que requiera de una interfaz cerebro-ordenador.

5. Bibliografía

Silver D., Thomas H. (2017) *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.*

Karras T., Laine S. (2019) *A Style-Based Generator Architecture for Generative Adversarial Networks.*

Chiuzbaian A, Jakobsen J. (2019) *Mind Controlled Drone: An Innovative Multiclass SSVEP based Brain Computer Interface.*

Bustamante E. (2007) *El sistema nervioso. desde las neuronas hasta el cerebro humano.*

Niedermeyer E., Da Silva F. (2005) *Electroencephalography: basic principles, clinical applications, and related fields.*

Walker P. (1999) *Chambers dictionary of science and technology.*

Deuschl G., Eisen A. (1999) *Recommendations for the practice of clinical neurophysiology: guidelines of the International Federation of Clinical Neurophysiology.*

Baumeister J, Barthel T. (2008) *Influence of phosphatidylserine on cognitive performance and cortical activity after induced stress.*

Christopher M. (1996) *Neural Networks for Pattern Recognition*

Google LLC, *Keras Documentation* (https://keras.io/)

Freed A., Schmeder A. (2009) *Features and Future of Open Sound Control version 1.1 for NIME Conference*

6. Anexos

6.1. Hoja de especificaciones de Muse

TECHNICAL SPECIFICATIONS

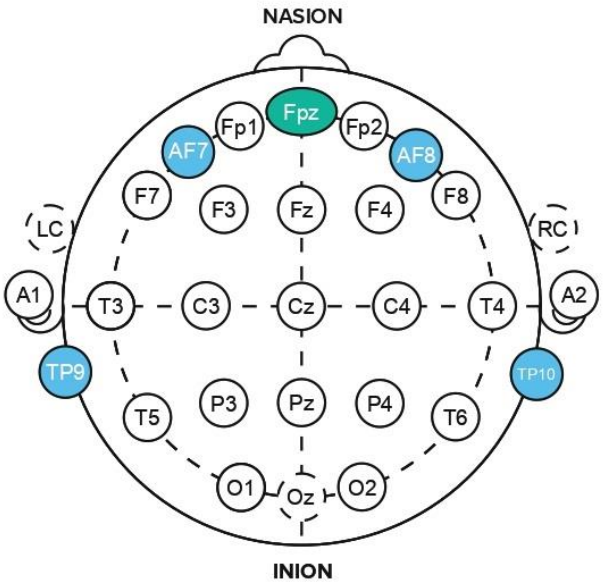
TECHNICAL SPECIFICATIONS

WIRELESS CONNECTION	Bluetooth 2. 1 + EDR	
SAMPLE RATE	220Hz or 500Hz, user selectable	
REFERENCE ELECTRODE	FPz (CMS/DRL)	
CHANNEL ELECTRODES	TP9, AF7, AF8, TP10 (dry)	
BATTERY LIFE	Maximum 5 hours (rechargeable Li -Ion)	
MATERIALS	Silver (frontal electrodes), Conductive silicone -rubber (temporal electrodes)	
WEIGHT	61g	
DIMENSIONS	Minimum head circumference	52cm
	Maximum head circumference	60cm
ACCELEROMETER	Three-axis @ 50Hz, 10 bit resolution, range +/ - 2G	
INPUT RANGE	2mV p-p AC coupled signal	
NOISE SUPPRESSION	DRL – REF feedback with 2µV (RMS) noise floor	
	50 or 60Hz notch filter (regional)	
MUSE APP COMPATIBILITY	iOS, Android	
RESEARCH TOOL COMPATIBILITY	Windows, Mac OS, Linux	
LIBMUSE COMPATIBILITY	iOS, Android, Windows	

Muse electrode locations by 10-20 International Standards.

Reference

Channel



6.2. Código de MuseUtils

```
import os
from threading import Thread
import numpy as np
from pythonosc.osc_server import ThreadingOSCUDPServer
from pythonosc.dispatcher import Dispatcher
from time import sleep
class MuseReceiver:
    def __init__(self, ip = "127.0.0.1", port = 1337):
        # Receiver parameters
        self.ip = ip
        self.port = port
        print(self.port)
        self.EEGpath = "data"
        self.listening = False
        # Received data
        self.battery = 0
        self.EEGsamples = -1
        self.EEGsample = 0
    def startServer(self):
        if not self.listening:
            # Pairing data to its respective processing function
            disp = Dispatcher()
            disp.map("/batt", self.updateBattery)
            disp.map("/eeg", self.updateEEG)
            # Starting OSC's receiver server
            self.server = ThreadingOSCUDPServer((self.ip, self.port), disp
)
            Thread(target = self.server.serve_forever).start()
            print("Receiving Muse data from " + self.server.server_address
[0]
            + " over port " + str(self.server.server_address[1]) + ".")
            print(self.server.server_address[1])
            self.listening = True
    def stopServer(self):
        self.server.shutdown()
        self.server = None
        self.listening = False
        print("Muse receiver stopped working.")
    def updateBattery(self, address, value, *_):
        self.battery = round(value, 2)
    def getBattery(self):
        return self.battery
    def updateEEG(self, address, channel1, channel2, channel3, channel4, *
_):
        if self.EEGsample < self.EEGsamples:
```

```
            # Storing each channel's sample
            self.EEG[self.EEGsample][:] = np.array([channel1, channel2, ch
annel3, channel4])
        if self.EEGsample == self.EEGsamples:
            # Saving snap on a CSV file
            if not os.path.exists(self.EEGpath):
                try:
                    os.makedirs(self.EEGpath)
                except:
                    pass
            np.savetxt(self.EEGpath + "/" + self.EEGname, self.EEG, delimi
ter=",")
        self.EEGsample += 1
    def EEGsnap(self, ms = 100, filename = "default"):
        self.EEGpath = 'data/' + '/'.join(filename.split("/")[:-1])
        self.EEGname = filename.split("/")[-1] + ".csv"
        self.EEGsamples = round(ms/1000*250)
        self.EEGsample = 0
        self.EEG = np.zeros((self.EEGsamples, 4))
```

6.3. Código de DatasetMaker

```
import sys
import tkinter as tk
import tkinter.ttk as ttk
from threading import Thread
from time import sleep
from MuseUtils import MuseReceiver
import datetime
from time import sleep
import numpy as np

import matplotlib.backends.tkagg as tkagg
from matplotlib.backends.backend_agg import FigureCanvasAgg
import matplotlib as mpl

def on_closing():
    if top.running:
        top.Disconnect()
    root.destroy()

class MainWindow:
    def Connect(self):
        if not self.running:
            try:
                self.receiver = MuseReceiver(self.ipEntry.get(), int(self.
portEntry.get()))
                self.receiver.startServer()
                self.running = True
                self.ConnectButtonText.set("Stop")
                Thread(target = self.updateBattery).start()
            except:
                pass
        else:
            self.ConnectButtonText.set("Start")
            self.receiver.stopServer()
            self.running = False

    def Sample(self):
        if self.running:
            self.receiver.EEGsnap(int(self.SampleDurEntry.get()), self.sub
jectEntry.get() + "/" + datetime.datetime.now().strftime("%Y-%m-
%d") + "/" + self.SampleNameEntry.get())
            sleep(int(self.SampleDurEntry.get())/1000+1)
            EEG = np.genfromtxt(self.receiver.EEGpath + "/" + self.receive
r.EEGname, delimiter=',')
```

```
            print(EEG[:].shape)
            import matplotlib.pyplot as plt
            fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, sharex=True)
            ax1.plot(EEG[:, 0])
            ax2.plot(EEG[:, 1])
            ax3.plot(EEG[:, 2])
            ax4.plot(EEG[:, 3])
            fig.show()
            #plt.show()
    def Disconnect(self):
        self.receiver.stopServer()
        self.running = False
    def updateBattery(self):
        while self.running:
            sleep(1)
            try:
                self.progress.set(round(self.receiver.getBattery(), 1))
                self.batteryLabel.config(text='Battery ' + str(self.receiv
er.getBattery()) + "%")
            except:
                pass

    def __init__(self, top=None):
        self.running = False
        top.geometry("894x660+284+31")
        top.title("Dataset Maker")

        self.Title = tk.Label(top)
        self.Title.configure(text="Dataset Maker 0.1")
        self.Title.configure(font="-family {Segoe UI} -size 24 -
weight bold -slant roman -underline 0 -overstrike 0")
        self.Title.place(relx=0.078, rely=0.045, height=71, width=264)

        self.Channel0 = tk.Canvas(top)
        self.Channel0.place(relx=0.459, rely=0.076, relheight=0.202, relwi
dth=0.473)
        self.Channel0.configure(relief='ridge')
        self.Channel0.configure(borderwidth="2")

        self.Channel1 = tk.Canvas(top)
        self.Channel1.place(relx=0.459, rely=0.303, relheight=0.202, relwi
dth=0.473)
        self.Channel1.configure(relief='ridge')
        self.Channel1.configure(borderwidth="2")

        self.Channel3 = tk.Canvas(top)
        self.Channel3.place(relx=0.459, rely=0.758, relheight=0.202, relwi
dth=0.473)
        self.Channel3.configure(relief='ridge')
```

```
        self.Channel3.configure(borderwidth="2")

        self.Channel2 = tk.Canvas(top)
        self.Channel2.place(relx=0.459, rely=0.53, relheight=0.202, relwid
th=0.473)
        self.Channel2.configure(relief='ridge')
        self.Channel2.configure(borderwidth="2")

        self.ReceiverFrame = tk.LabelFrame(top)
        self.ReceiverFrame.configure(text="Receiver options")
        self.ReceiverFrame.place(relx=0.022, rely=0.182, relheight=0.159,
relwidth=0.403)

        self.ipLabel = tk.Label(self.ReceiverFrame)
        self.ipLabel.configure(text="IP:")
        self.ipLabel.configure(anchor='w')
        self.ipLabel.place(relx=0.056, rely=0.286, height=21, width=34, bo
rdermode='ignore')

        self.portLabel = tk.Label(self.ReceiverFrame)
        self.portLabel.configure(text="Port:")
        self.portLabel.configure(anchor='w')
        self.portLabel.place(relx=0.056, rely=0.571, height=21, width=34,
bordermode='ignore')

        self.ipEntry = tk.Entry(self.ReceiverFrame)
        self.ipEntry.place(relx=0.167, rely=0.286, height=20, relwidth=0.3
17, bordermode='ignore')

        self.portEntry = tk.Entry(self.ReceiverFrame)
        self.portEntry.place(relx=0.167, rely=0.571, height=20, relwidth=0
.122, bordermode='ignore')

        self.ConnectButtonText = tk.StringVar()
        self.ConnectButtonText.set("Start")
        self.ConnectButton = tk.Button(self.ReceiverFrame)
        self.ConnectButton.configure(command=self.Connect)
        self.ConnectButton.configure(textvariable=self.ConnectButtonText)
        self.ConnectButton.place(relx=0.333, rely=0.571, height=24, width=
56, bordermode='ignore')

        self.progress = tk.IntVar()
        self.TProgressbar1 = ttk.Progressbar(self.ReceiverFrame, variable
= self.progress)
        self.TProgressbar1.place(relx=0.583, rely=0.571, relwidth=0.361, r
elheight=0.0, height=22, bordermode='ignore')

        self.batteryLabel = tk.Label(self.ReceiverFrame)
        self.batteryLabel.configure(text='''Battery: ?%''')
```

```
        self.batteryLabel.place(relx=0.583, rely=0.286, height=21, width=1
00, bordermode='ignore')

        self.ReceiverSeparator = ttk.Separator(self.ReceiverFrame)
        self.ReceiverSeparator.place(relx=0.528, rely=0.143, relheight=0.7
62, bordermode='ignore')
        self.ReceiverSeparator.configure(orient="vertical")

        self.TestFrame = tk.LabelFrame(top)
        self.TestFrame.configure(text="Test info")
        self.TestFrame.place(relx=0.112, rely=0.409, relheight=0.508, relw
idth=0.213)

        self.SubjectLabel = tk.Label(self.TestFrame)
        self.SubjectLabel.place(relx=0.105, rely=0.239, height=21, width=7
2, bordermode='ignore')
        self.SubjectLabel.configure(text="Test subject:")

        self.subjectEntry = tk.Entry(self.TestFrame)
        self.subjectEntry.place(relx=0.105, rely=0.299, height=20, relwidt
h=0.758, bordermode='ignore')

        self.DateLabel = tk.Label(self.TestFrame)
        self.DateLabel.configure(text="Test date: " + datetime.datetime.no
w().strftime("%d-%m-%Y"))
        self.DateLabel.place(relx=0.105, rely=0.09, height=21, width=118,
bordermode='ignore')

        self.SampleNameLabel = tk.Label(self.TestFrame)
        self.SampleNameLabel.configure(text="Samle name:")
        self.SampleNameLabel.place(relx=0.105, rely=0.448, height=21, widt
h=74, bordermode='ignore')

        self.SampleNameEntry = tk.Entry(self.TestFrame)
        self.SampleNameEntry.place(relx=0.105, rely=0.507, height=20, relw
idth=0.758, bordermode='ignore')

        self.SampleDurLabel = tk.Label(self.TestFrame)
        self.SampleDurLabel.configure(text='''Sample duration (ms):''')
        self.SampleDurLabel.place(relx=0.105, rely=0.657, height=21, width
=123, bordermode='ignore')

        self.SampleDurEntry = tk.Entry(self.TestFrame)
        self.SampleDurEntry.place(relx=0.105, rely=0.716, height=20, relwi
dth=0.758, bordermode='ignore')

        self.SampleButton = tk.Button(self.TestFrame)
        self.SampleButton.configure(text="Sample!")
        self.SampleButton.configure(command=self.Sample)
```

```
        self.SampleButton.place(relx=0.105, rely=0.866, height=24, width=1
47, bordermode='ignore')

receiver = None
root = tk.Tk()
top = MainWindow(root)

if __name__ == '__main__':
    root.protocol("WM_DELETE_WINDOW", on_closing)
    root.mainloop()
```

6.4. Código del receptor de chat de Telegram

```
import os, sys, json, datetime
import utils
from random import randint

utils.cls()

bot_name, token, ownerID, command_char = utils.get_bot_config()
bot = telebot.TeleBot(token)

import screen
app = screen.App()
def showMessage(m):
    if m.text == "/clear":
        words = ["Perrito", "Gatito", "Plens", "Fourier",
        "Laplace", "Gauss", "John Travolta", "A::Microtech",
        "Blanco", "Patatas", "Telegram", "WhatsApp", "Facebook",
        "Instagram", "Twitter", "Disney", "Warner Bros.", "Mario Bros.",
        "Luigi Bros.", "Peach", "Wario", "Yoshi", "Pikachu",
        "Squirtle", "Charmander", "Mudkip", "Dragonite"]
        app.updatetext(palabras[randint(0, len(words)-1)])
    else:
        app.updatetext(m.text)

def listener(messages):
    for m in messages:
        showMessage(m)

try:
    bot.send_message(ownerID, bot_name + ' is now running.')
    print(bot_name + colored(' is running!', 'green'))
    bot.set_update_listener(listener)
    bot.polling(True)
except:
    print(colored('''Can't start the bot, maybe is not connected to intern
et.''', 'red'))
```

6.5. Código de la aplicación auxiliar

```
import tkinter as tk
import threading

class App(threading.Thread):

    def __init__(self):
        threading.Thread.__init__(self)
        self.v = None
        self.start()

    def callback(self):
        self.root.quit()

    def run(self):
        self.root = tk.Tk()
        self.root.configure(background='black')
        self.root.attributes("-fullscreen", True)
        self.root.protocol("WM_DELETE_WINDOW", self.callback)
        self.v = tk.StringVar()
        self.v.set("¡Bienvenid@!")
        self.label = tk.Label(self.root, textvariable=self.v, font=("Helve
tica", 72))
        self.label.grid(column=0, row=0)
        self.label.config(fg="lightgreen", bg="black")
        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
        self.root.mainloop()

    def updatetext(self, text):
        self.v.set(text)
```

6.6. Código de la red con mayor rendimiento

```
import sys, os
from keras import optimizers
from keras.models import Sequential
from keras.layers import InputLayer, Convolution2D, MaxPooling2D, UpSampli
ng2D, Dense, Dropout, Flatten, Activation
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import plot_model

# Data parameters
img_width, img_height = 600, 536
train_data_path = 'data/train'
validation_data_path = 'data/test'
classes_num = 4
# Training parameters
epochs = 20
batch_size = 8
samples_per_epoch = 1000
validation_steps = 300
# Model parameters
pool_size = 5
lr = 0.001
# Model implementation
model = Sequential()
model.add(InputLayer(input_shape=(img_height, img_width, 3)))

model.add(Convolution2D(10, 5, 5))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

model.add(Convolution2D(10, 5, 5))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(classes_num, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer=optimizers.RMSpro
p(lr=lr), metrics=['accuracy'])
```

```
# Preparing data for training
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(train_data_path, targe
t_size=(img_height, img_width), batch_size=batch_size, class_mode='categor
ical')
test_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = test_datagen.flow_from_directory(validation_data_pa
th, target_size=(img_height, img_width), batch_size=batch_size, class_mode
='categorical')
# Training model
model.fit_generator(train_generator, samples_per_epoch=samples_per_epoch,
epochs=epochs, validation_data=validation_generator, validation_steps=vali
dation_steps)
```